INITIAL PROJECT REPORT

*Dissertation submitted in fulfilment of the requirements for the Degree of*

# BACHELOR OF TECHNOLOGY

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**– DATA SCIENCE WITH MACHINE LEARNING**

By

**Panta Jayanth Reddy**

**Registration No: 12210770**

Supervisor

**SHUBHAM SHARMA**



**School of Computer Science and Engineering**

Lovely Professional University Phagwara, Punjab (India) October 2023

# DECLARATION STATEMENT

I hereby declare that the work reported in the Assignment Project entitled "PRIORITY SCHEDULING OF PROCESSES” in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering – Data Science with Machine Learning at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Shubham Sharma. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University’s Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

**Panta Jayanth Reddy**

**R. No: 12210770**

# TABLE OF CONTENTS

1. Introduction
2. Objectives And Scope of Project
3. Methodology
4. Flowchart
5. Summary

# INTRODUCTION

In this era of computer systems and operating environments, ensuring efficient and fair utilization of computational resources is imperative. One of the critical resources that need efficient management is the Central Processing Unit (CPU). To this end, operating systems deploy scheduling algorithms to decide which process in the ready queue should be assigned to the CPU next. Among these algorithms, Priority Scheduling stands out for its approach in determining the sequence based on process priorities.

Priority Scheduling is a preemptive or non-preemptive scheduling algorithm that assigns a priority to each process. The process with the highest priority (represented by the lowest numerical value, usually) is served first. If two processes have the same priority, then the scheduling decision falls back on a secondary criterion, like First Come First Serve (FCFS).

# OBJECTIVES AND SCOPE OF THE PROJECT

Objectives:

* Design and implement a priority scheduling system that maximizes the CPU's active processing time, ensuring it's rarely idle when there are tasks to be performed.
* Minimize the average time taken for processes to complete from the time of submission to the time of completion.
* Implement dynamic priority assignment techniques, such as aging, to adjust the priorities of processes over time, ensuring that long-waiting processes gradually get higher priorities.
* Design the scheduling system in a way that it can be easily integrated into existing operating systems or computing environments without significant delays.
* Allow system administrators or users to adjust priority rules and parameters easily, providing flexibility in changing the system's behavior as needs evolve..
* If the project involves user interaction, design an intuitive interface that allows users to set or adjust process priorities easily and view the status of their tasks.
* Ensure that the scheduling system can handle unexpected conditions (e.g., process crashes) without causing system-wide failures.

Scope:

Priority scheduling, as a principle, extends far beyond just the realm of computer processes. It is a concept rooted in efficiently allocating limited resources to tasks or entities based on their importance, urgency, or any other criteria deemed relevant. The scope for priority scheduling in the broader world is vast. Here are some areas where priority scheduling principles can be applied:

* Healthcare: Systems in emergency rooms determine the order in which patients are treated based on the severity of their conditions. Similarly, organ transplant lists prioritize recipients based on several factors including the urgency of need and expected outcomes.
* Transportation:
* Air Traffic Control: Planes are scheduled for takeoff, landing, and even in-flight pathways based on priority determined by factors like fuel levels, emergency situations, and scheduled timelines.
* Public Transit: During peak hours, extra buses or trains might be scheduled to accommodate the influx of passengers.
* Manufacturing: Assembly lines can be scheduled based on priority orders or the importance of certain products during specific seasons or promotional periods.
* Urban Planning: Development projects in cities might be prioritized based on immediate needs, such as repairing deteriorating infrastructure versus creating a new park.
* Energy Distribution: During power shortages, electricity might be allocated based on priority needs, ensuring essential services like hospitals and emergency services have power.
* Telecommunications: Data packets on a network might be prioritized based on the type of data (e.g., emergency calls over regular data packets) or Quality of Service (QoS) configurations.
* Financial Services: Loan or grant applications can be processed based on priority criteria, such as urgency, amount, or potential impact on economic development.
* Education: Resources like lab time, library resources, or counseling sessions can be scheduled based on students' needs and urgency.
* Event Management: Priority can be given to specific events or parts of an event based on importance, potential revenue, or other factors.
* Emergency Services: In disaster management, resources are often scarce. Prioritizing their distribution (food, water, shelter) can be based on the number of people affected, the severity of their situation, or other critical factors.
* Environmental Initiatives: Projects aimed at conservation or rehabilitation might be scheduled based on factors like the level of environmental threat, potential impact, or available resources.

In essence, any situation or sector that deals with limited resources and a multitude of tasks or requirements can benefit from priority scheduling. The objective remains consistent: ensuring resources are used effectively to achieve the most significant possible impact or outcome.

# METHODOLOGY

# The methodology for implementing priority scheduling for processes is a structured approach that ensures efficient and effective execution. The following steps provide a detailed methodology for designing and implementing priority scheduling:

# Requirement Analysis:

# Gather Requirements: Understand the specific requirements of the system where priority scheduling will be implemented. This includes the number of processes, the frequency of their arrival, constraints, and any specific goals.

# Design:

# Choose Scheduling Type: Decide whether the scheduling will be preemptive (where a running process can be interrupted by a higher-priority process) or non-preemptive (where processes run to completion regardless of incoming process priorities).

# Address Starvation: Design mechanisms to prevent starvation of low-priority processes. A common method is aging, where processes increase in priority as they wait.

# Initialize Data Structures: Start with an empty process queue or list.

# Load Processes: As processes arrive, assign them a priority and place them in the appropriate position in the queue or list based on their priority.

# Preemptive: Begin with the highest priority process. If a higher-priority process arrives, interrupt the current process, save its state, and start the higher-priority process.

# Testing and Validation:

# Unit Testing: Test individual components or functions of the scheduling system to ensure they work as intended.

# Integration Testing: Ensure that all components work together seamlessly.

# Performance Testing: Measure metrics like average waiting time, average turnaround time, and CPU utilization to evaluate the efficiency of the priority scheduling system.

# Address Anomalies: If any anomalies or inefficiencies are found, adjust the scheduling logic or design and retest.

# Optimization:

# Tweak Priority Assignments: Adjust how priorities are assigned based on performance outcomes or changing requirements.

# Adjust for Scalability: If the system needs to handle a larger number of processes, ensure the priority scheduling mechanism can scale without significant performance degradation.

# Deployment and Monitoring:

# Deploy: Integrate the priority scheduling mechanism into the target system or environment.

# Maintenance and Iteration:

# Feedback Loop: Establish a feedback mechanism to gather insights on the performance of the scheduling system.

# Updates and Iterations: Based on feedback and changing requirements, make necessary updates or improvements to the priority scheduling logic or implementation.

# This methodology offers a comprehensive approach to implementing priority scheduling in a systematic and effective manner. It's essential to continuously evaluate and iterate based on real-world performance and feedback.

# FLOWCHART:

The flowchart of the project illustrates the step-by-step process from inputting the processes to their execution of Priority Scheduling of Processes.

**Algorithm Implementation (Pseudocode):**

BEGIN

Structure PROCESS:

id: integer

priority: integer

burstTime: integer

arrivalTime: integer

waitTime: integer = 0

turnaroundTime: integer = 0

List<PROCESS> processList

List<PROCESS> readyQueue

List<PROCESS> completedQueue

Function SortByPriorityAndArrival(readyQueue):

Sort readyQueue first by priority (higher value indicates higher priority)

If two processes have the same priority, sort them by arrivalTime.

MAIN:

// Initialization

Load processes into processList

currentTime = 0

WHILE processList is not empty OR readyQueue is not empty:

// Add processes to the readyQueue if their arrivalTime <= currentTime

FOR each process in processList:

IF process.arrivalTime <= currentTime:

Add process to readyQueue

Remove process from processList

IF readyQueue is not empty:

SortByPriorityAndArrival(readyQueue)

currentProcess = first item in readyQueue

currentProcess.waitTime = currentTime - currentProcess.arrivalTime

currentProcess.turnaroundTime = currentProcess.waitTime + currentProcess.burstTime

currentTime += currentProcess.burstTime

Move currentProcess from readyQueue to completedQueue

ELSE:

// If no processes are in the readyQueue, increment the currentTime

currentTime += 1

END WHILE

Display completedQueue or calculate average wait and turnaround times as needed

END

# SUMMARY

Priority scheduling is an operating system technique where processes are executed based on assigned priority levels. Each process receives a priority, either statically or dynamically. In preemptive priority scheduling, a higher-priority process can interrupt a running lower-priority process. In non-preemptive mode, a process completes its execution irrespective of the priority of incoming processes. Solutions like "aging" adjust priorities over time to combat this. This scheduling approach ensures crucial tasks receive timely CPU attention, optimizing system responsiveness and efficiency.